What makes a model?

Fitting Models

How do you fit a linear model in R?
How many different ways can you think of?

- `lm` for linear model
- `glm` for generalized linear model (e.g. logistic regression)
- `glmnet` for regularized regression
- `keras` for regression using TensorFlow
- `stan` for Bayesian regression
- `spark` for large data sets

To specify a model

- Choose a model
- Specify an engine
- Set the mode

To specify a model

```
library(tidymodels)
linear_reg()
#> Linear Regression Model Specification (regression)
#>
#> Computational engine: lm
```

To specify a model

```
linear_reg() %>%
  set_engine("glmnet")
#> Linear Regression Model Specification (regression)
#>
#> Computational engine: glmnet
```

To specify a model

```
linear_reg() %>%
  set_engine("stan")
#> Linear Regression Model Specification (regression)
#>
#> Computational engine: stan
```

To specify a model

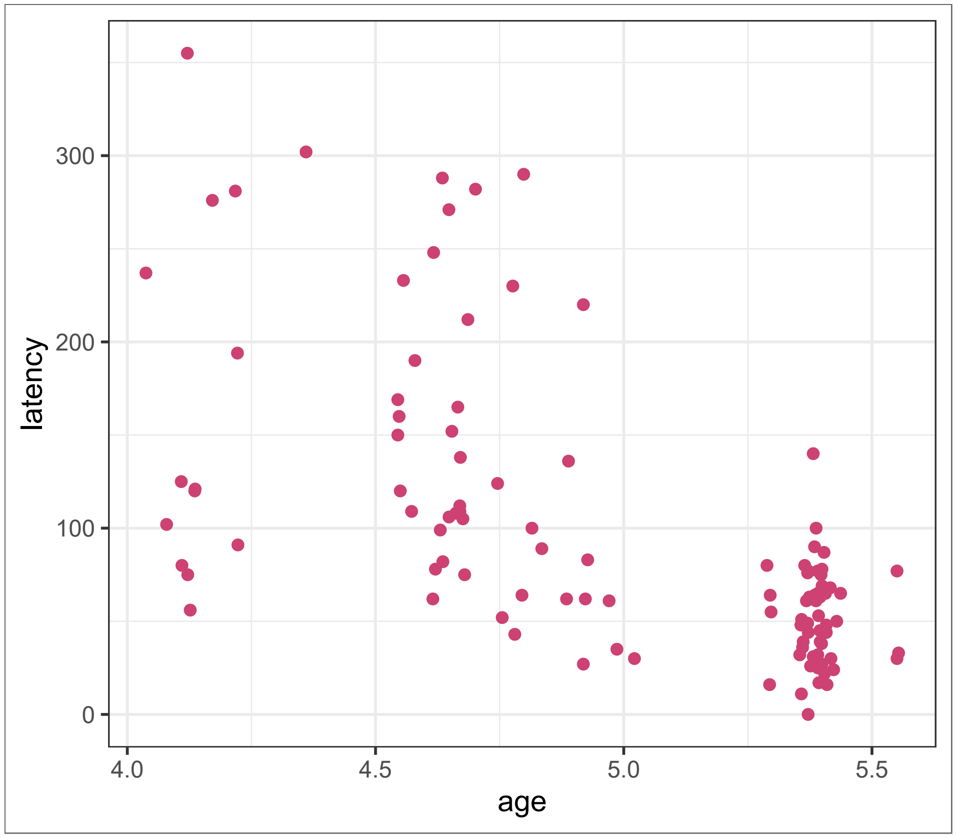
```
decision_tree()
#> Decision Tree Model Specification (unknown mode)
#>
#> Computational engine: rpart
```

To specify a model

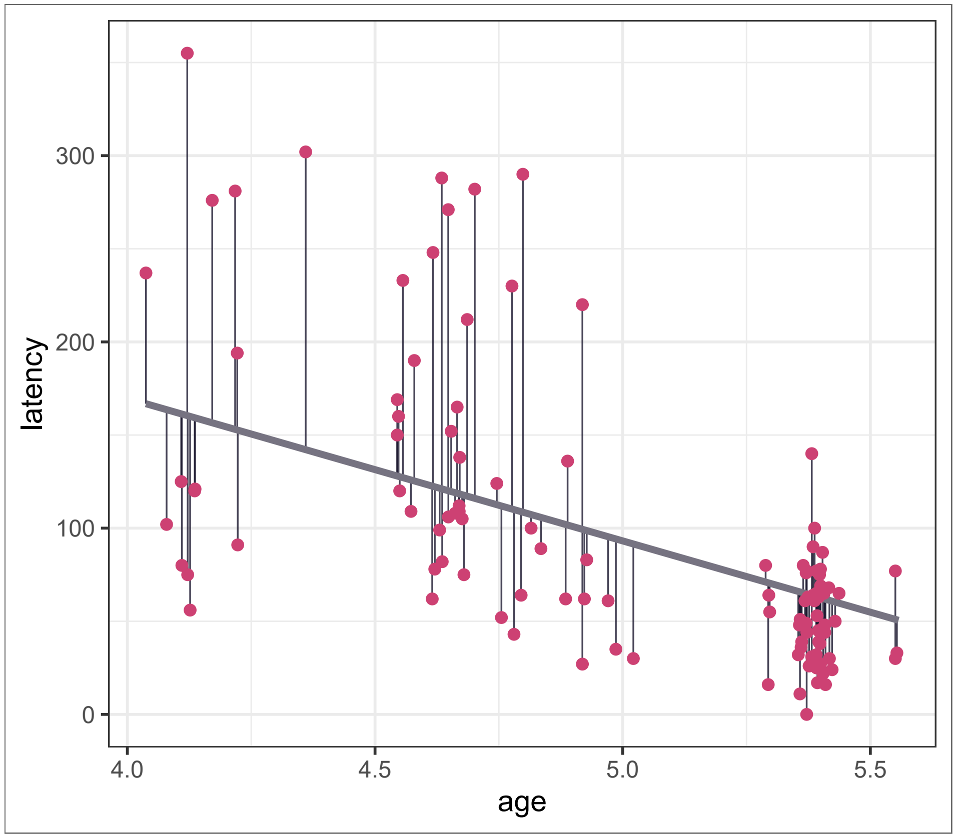
```
decision_tree() %>%
  set_mode("regression")
#> Decision Tree Model Specification (regression)
#>
#> Computational engine: rpart
```

All available models are listed at https://www.tidymodels.org/find/parsnip/

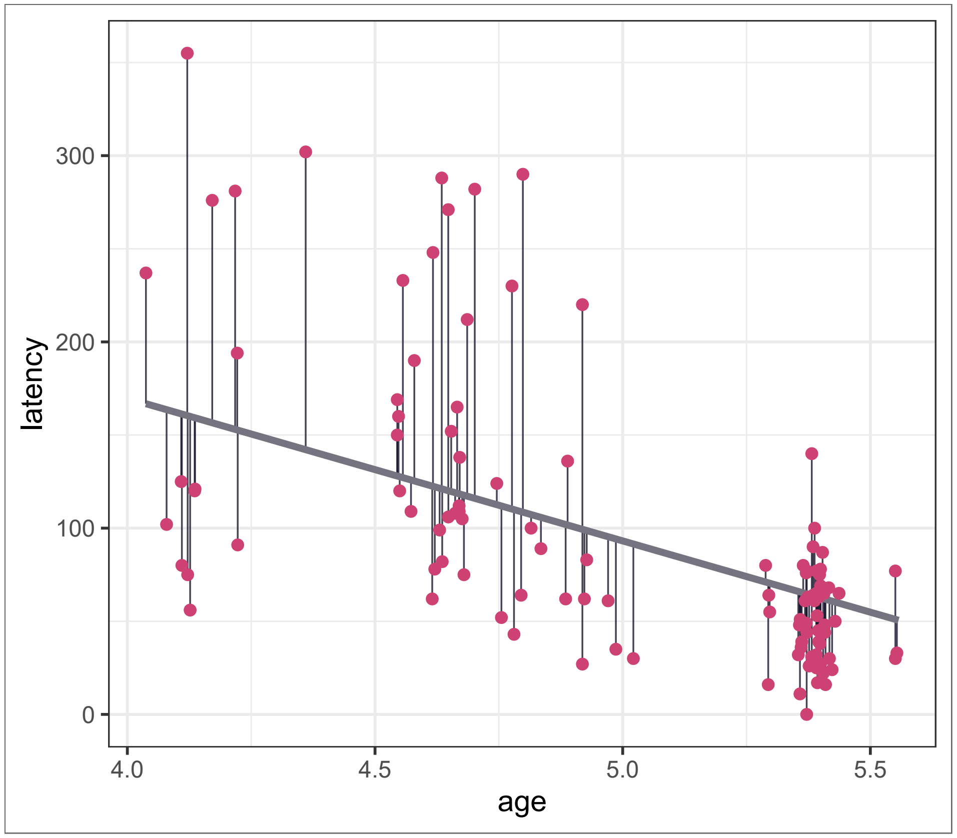
Linear regression



Linear regression



Linear regression

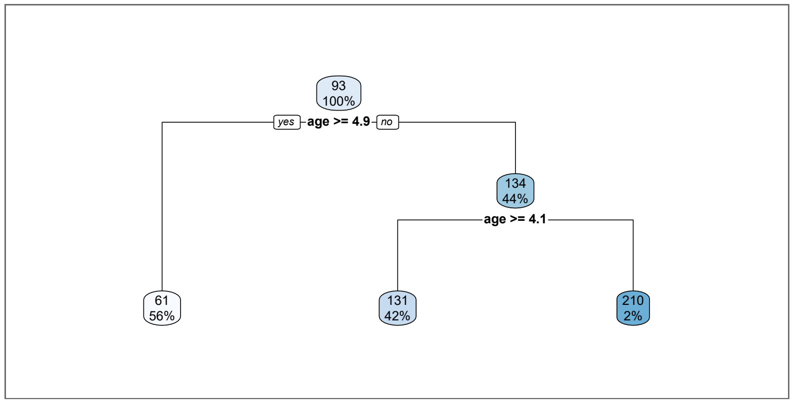


- Outcome modeled as linear combination of predictors:

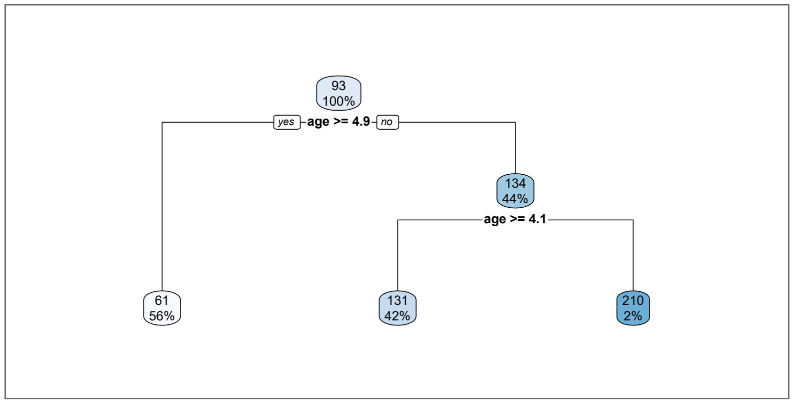
$$\text{latency} = \beta_0 + \beta_1 \cdot \text{age} + \epsilon$$

- Find a line that minimizes the mean squared error (MSE)

Decision trees

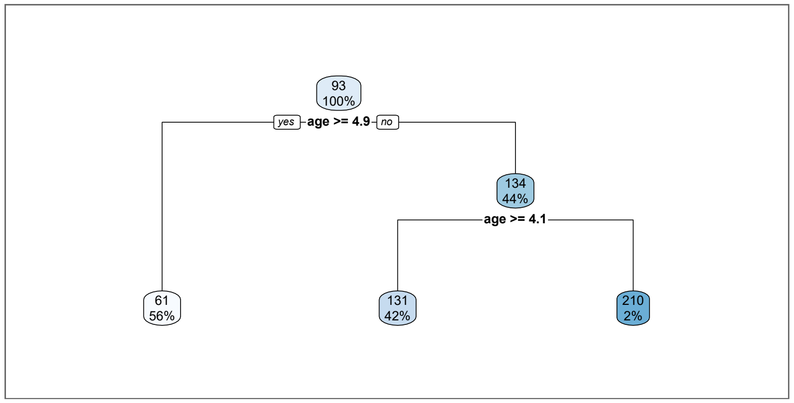


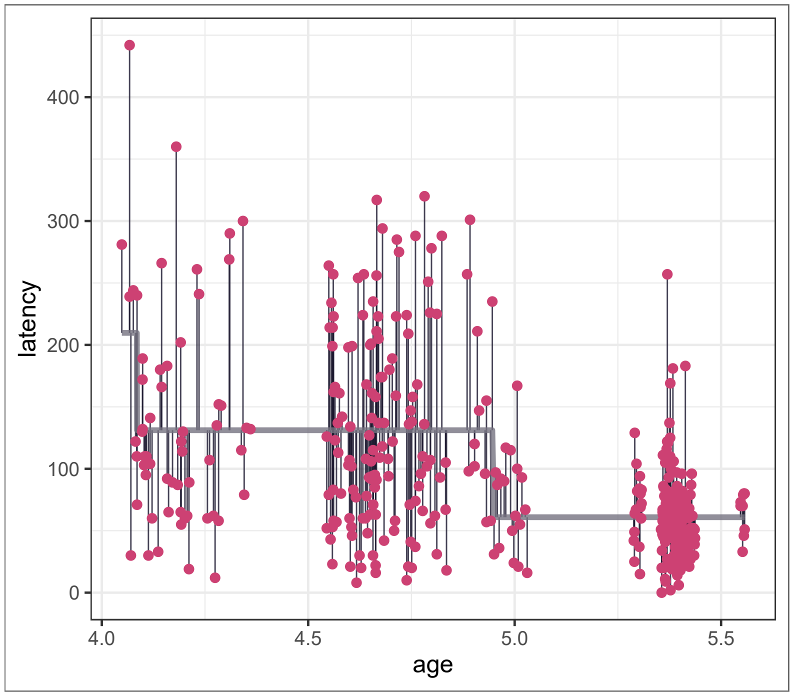
Decision trees



- Series of splits or if/then statements based on predictors
- First the tree *grows* until some condition is met (maximum depth, no more data)
- Then the tree is *pruned* to reduce its complexity

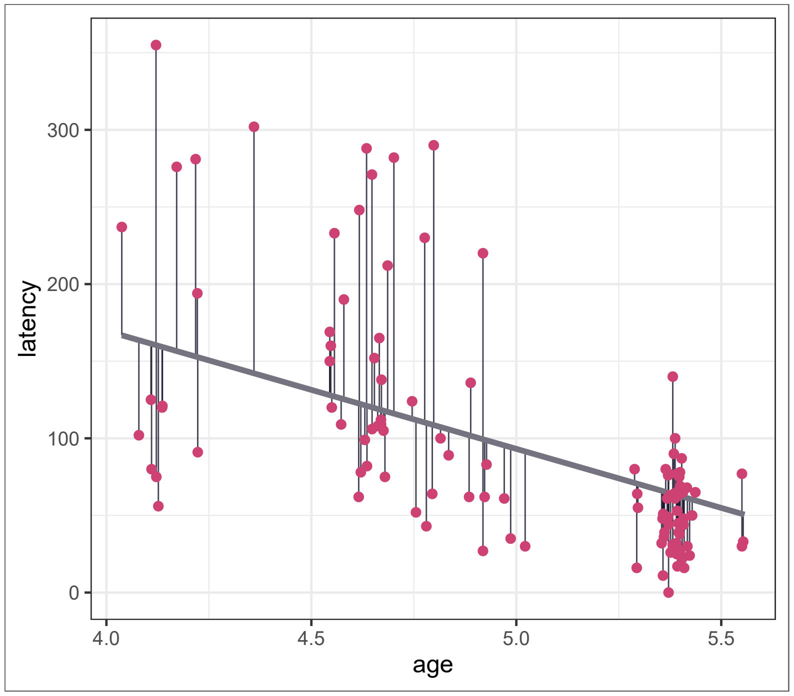
Decision trees



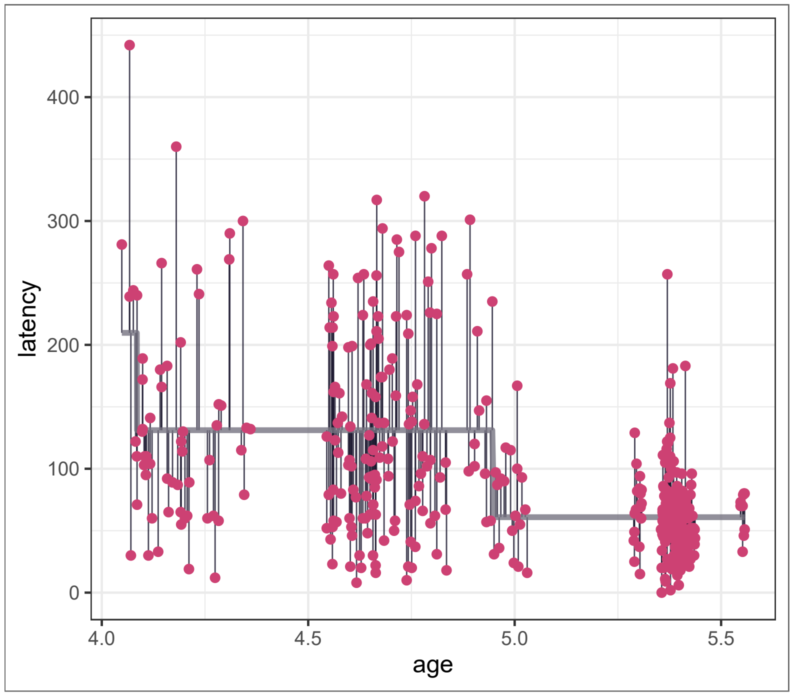


All models are wrong, but some are useful!

Linear regression

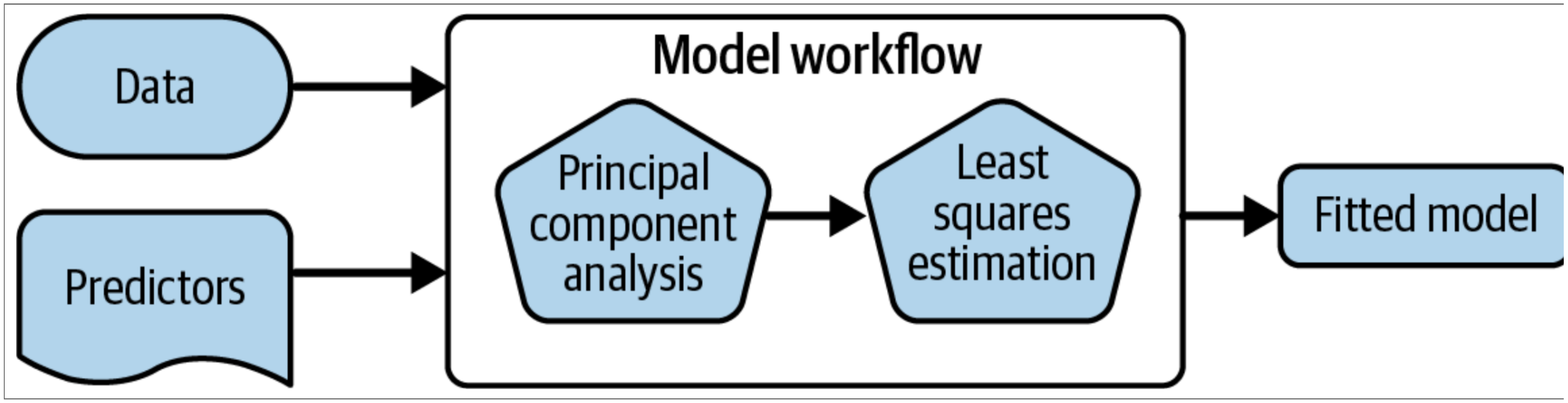


Decision trees

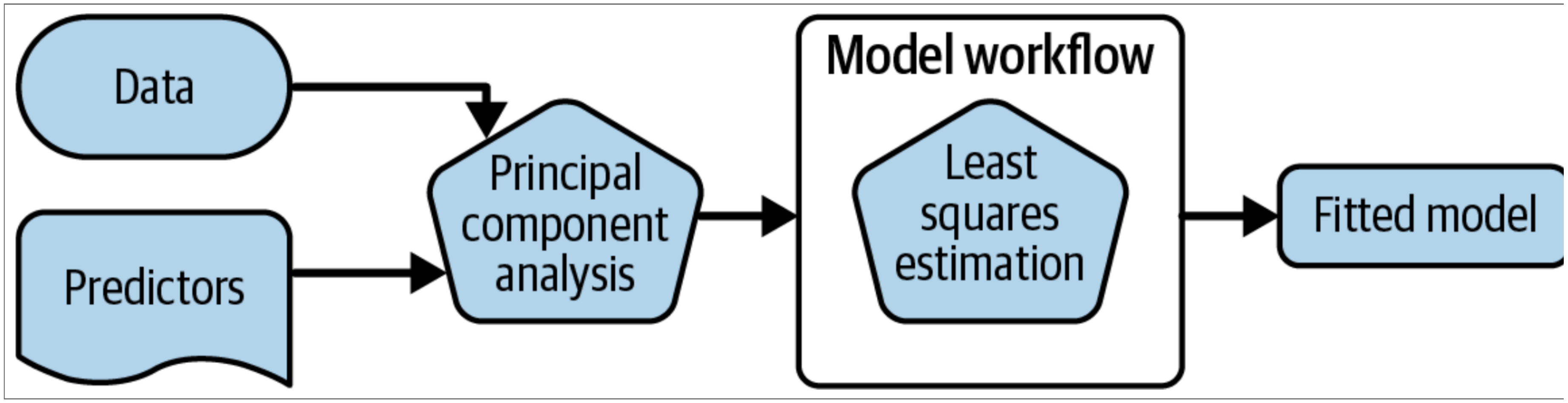

latency
400
300
200
100
0
4.0
4.5
5.0
5.5
age

A model workflow

Workflows bind preprocessors and models



What is wrong with this?



Why a `workflow()`?

- Workflows handle new data better than base R tools in terms of new factor levels
- You can use other preprocessors besides formulas (more on feature engineering tomorrow!)
- They can help organize your work when working with multiple models
- Most importantly, a workflow captures the entire modeling process: `fit()` and `predict()` apply to the preprocessing steps in addition to the actual model fit

A model workflow

```r
tree_spec <-
  decision_tree() %>%
  set_mode("regression")

tree_spec %>%
  fit(latency ~ ., data = frog_train)
#> parsnip model object
#>
#> n= 456
#>
#> node), split, n, deviance, yval
#>       * denotes terminal node
#>
#>  1) root 456 2197966.00  92.90351
#>    2) age>=4.947975 256  252347.40  60.89844
#>      4) treatment=control 131   91424.06  48.42748 *
#>      5) treatment=gentamicin 125  119197.90  73.96800 *
#>    3) age< 4.947975 200 1347741.00 133.87000
#>      6) treatment=control 140  986790.70 118.25710
```

A model workflow

```r
tree_spec <-
  decision_tree() %>%
  set_mode("regression")

workflow() %>%
  add_formula(latency ~ .) %>%
  add_model(tree_spec) %>%
  fit(data = frog_train)
#> ══ Workflow [trained] ══════════════════════════════════════════════════
#> Preprocessor: Formula
#> Model: decision_tree()
#>
#> ── Preprocessor ────────────────────────────────────────────────────────
#> latency ~ .
#>
#> ── Model ───────────────────────────────────────────────────────────────
#> n= 456
#>
#> node), split, n, deviance, yval
```

A model workflow

```r
tree_spec <-
  decision_tree() %>%
  set_mode("regression")

workflow(latency ~ ., tree_spec) %>%
  fit(data = frog_train)
#> ══ Workflow [trained] ══════════════════════════════════════════════════
#> Preprocessor: Formula
#> Model: decision_tree()
#>
#> ── Preprocessor ────────────────────────────────────────────────────────
#> latency ~ .
#>
#> ── Model ───────────────────────────────────────────────────────────────
#> n= 456
#>
#> node), split, n, deviance, yval
#>       * denotes terminal node
#>
```

Predict with your model

How do you use your new `tree_fit` model?

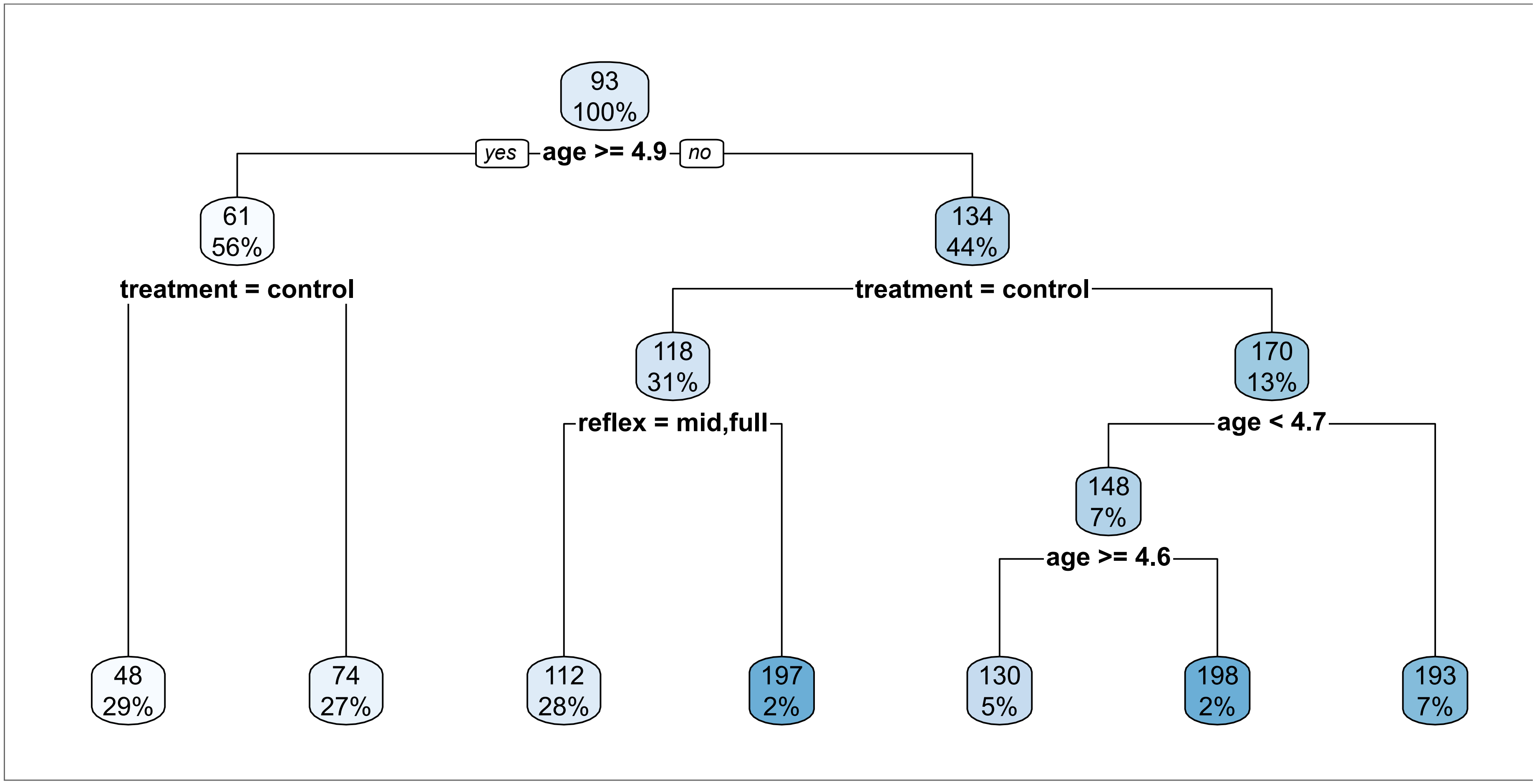
```r
tree_spec <-
  decision_tree() %>%
  set_mode("regression")

tree_fit <-
  workflow(latency ~ ., tree_spec) %>%
  fit(data = frog_train)
```

The tidymodels prediction guarantee!

- The predictions will always be inside a **tibble**
- The column names and types are **unsurprising** and **predictable**
- The number of rows in `new_data` and the output **are the same**

Understand your model

How do you **understand** your new `tree_fit` model?



Understand your model

How do you **understand** your new `tree_fit` model?

```r
library(rpart.plot)
tree_fit %>%
  extract_fit_engine() %>%
  rpart.plot(roundint = FALSE)
```

You can `extract_*()` several components of your fitted workflow.

Understand your model

How do you **understand** your new `tree_fit` model?

You can use your fitted workflow for model and/or prediction explanations:

- overall variable importance, such as with the **vip** package
- flexible model explainers, such as with the **DALEXtra** package

Learn more at **https://www.tmwr.org/explain.html**

Error

×